WHAT IS CLAIMED IS:

1. A method for dynamically modifying a stored program, the method comprising the steps of:

storing correction code in at least one of a plurality of correction blocks included in an electrically erasable programmable memory;

executing a program having instructions stored in the memory;

invoking an address match routine to execute at least a portion of the correction code in place of at least one of the instructions during the executing of the program; and

continuing executing the program after the at least a portion of the correction code executes.

2. The method of claim 1, wherein the step of invoking an address match routine occurs when a program counter associated with the executing of the program matches at least one of a plurality of address match registers.

3. The method of claim 2, further comprising the steps of:

determining for each of the correction blocks whether correction code stored in the blocks is to be executed during the executing of the program;

retrieving a first address from the correction code stored in a respective correction block when it is determined that correction code stored in the respective correction block is to be executed during program execution; and

storing the retrieved first address in one of the plurality of address match registers.

4. The method of claim 3, wherein the step of determining for each of the correction blocks whether correction code stored in a respective correction block is to be executed during program execution comprises the steps of:

retrieving a first data value from each of the correction blocks having stored correction code;

comparing the retrieved first data value from each of the correction blocks having stored correction code to a predetermined value; and

determining that the correction code stored in each of the correction blocks is to be executed during program execution when the corresponding first data value equals the predetermined value, otherwise determining that the correction code stored in each of the correction blocks is not to be executed during program execution.

5. The method of claim 2, wherein the step of determining for each of the correction blocks whether correction code stored in the blocks is to be executed during the executing of the program occurs in response to a completion of the storing of correction code in at least one of a plurality of correction blocks.

6. The method of claim 2, further comprising the steps of:

saving a plurality of registers and the program counter upon the invoking of the address match routine to preserve a post address match program status;

retrieving a second data value from each of the correction blocks having stored correction code;

comparing the retrieved second data value from each of the correction blocks having stored correction code to a post address match value of the program counter; and

identifying the correction block corresponding to the invoking of the address match routine as the correction block having the second data value equal to the post address match value of the program counter.

7. The method of claim 6, further comprising the step of:

branching to an error processing routine when no retrieved second data value is equal to the post address match value of the program counter.

8. The method of claim 6, wherein the second data value is equal to the value stored in the address match register corresponding to the invoking of the address match routine plus an offset value.

9. The method of claim 8, wherein the offset value is dependent upon a next program instruction to be executed at the time of the invoking of the address match routine.

10. The method of claim 6, wherein the step of continuing executing the program after the at least a portion of the correction code executes comprises the steps of:

retrieving a third address from the correction code identifying a return address within the program;

restoring the plurality of registers saved upon the invoking of the address match routine to reinstate a post address match program status;

branching the executing of the program to the third address; and

executing the program having instructions stored in the memory beginning at the third address.

11. The method of claim 1, wherein the invoking of the address match routine is carried out by an address match interrupt service routine having a corresponding address match interrupt entry in a vector table stored in the memory.

12. The method of claim 1, wherein to execute at least a portion of the correction code, the address match routine comprises the steps of:

retrieving a second address from the correction code identifying a starting address within the at least a portion of correction code;

branching the executing of the program to the second address; and

executing the at least a portion of the correction code beginning at the second address.

13. The method of claim 1, wherein instructions for continuing executing the program after the at least a portion of the correction code executes are included in the correction code.

14. The method of claim 1, wherein the step of storing correction code in at least one of a plurality of correction blocks comprises the steps of:

selecting at least one of the plurality of correction blocks for storing the correction code;

erasing the selected at least one memory correction block; and

transferring the correction code from an external source to the selected at least one of the plurality of correction blocks.

15. The method of claim 14, wherein the correction code is transferred from the external source by at least one of a wired and a wireless connection.

16. The method of claim 1, wherein the step of storing correction code in at least one of a plurality of correction blocks occurs in response to at least one of a detection that an external source is coupled to the memory and an invocation of a periodically scheduled maintenance routine.

17. A micro-controller capable of dynamically modifying a stored program, the micro-controller comprising:

electrically erasable programmable memory;

logic that stores correction code in at least one of a plurality of correction blocks included in the memory;

logic that executes a program having instructions stored in the memory;

logic that invokes an address match routine to execute at least a portion of the correction code in place of at least one of the instructions during the executing of the program; and

logic that continues executing the program after the at least a portion of the correction code executes.

18. The micro-controller of claim 17, wherein the logic that invokes an address match routine is responsive to a matching of a program counter associated with the executing of the program and at least one of a plurality of address match registers.

19. The micro-controller of claim 18, further comprising:

logic that determines for each of the correction blocks whether correction code stored in the blocks is to be executed during the executing of the program;

logic that retrieves a first address from the correction code stored in a respective correction block when it is determined that correction code stored in the respective correction block is to be executed during program execution; and

logic that stores the retrieved first address in one of the plurality of address match registers.

20. The micro-controller of claim 19, wherein the logic that determines for each of the correction blocks whether correction code stored in a respective correction block is to be executed during program execution comprises:

logic that retrieves a first data value from each of the correction blocks having stored correction code;

logic that compares the retrieved first data value from each of the correction blocks having stored correction code to a predetermined value; and

logic that determines that the correction code stored in each of the correction blocks is to be executed during program execution when the corresponding first data value equals the predetermined value, otherwise determining that the correction code stored in each of the correction blocks is not to be executed during program execution.

21. The micro-controller of claim 18, wherein the logic that determines for each of the correction blocks whether correction code stored in the blocks is to be executed during the executing of the program is responsive to a completion of the storing of correction code in at least one of a plurality of correction blocks.

22. The micro-controller of claim 18, further comprising:

logic that saves a plurality of registers and the program counter upon the invoking of the address match routine to preserve a post address match program status;

logic that retrieves a second data value from each of the correction blocks having stored correction code;

logic that compares the retrieved second data value from each of the correction blocks having stored correction code to a post address match value of the program counter; and

logic that identifies the correction block corresponding to the invoking of the address match routine as the correction block having the second data value equal to the post address match value of the program counter.

23. The micro-controller of claim 22, further comprising:

logic that branches to an error processing routine when no retrieved second data value is equal to the post address match value of the program counter.

24. The micro-controller of claim 22, wherein the second data value is equal to the value stored in the address match register corresponding to the invoking of the address match routine plus an offset value.

25. The micro-controller of claim 24, wherein the offset value is dependent upon a next program instruction to be executed at the time of the invoking of the address match routine.

26. The micro-controller of claim 22, wherein the logic that continues executing the program after the at least a portion of the correction code executes comprises:

logic that retrieves a third address from the correction code identifying a return address within the program;

logic that restores the plurality of registers saved upon the invoking of the address match routine to reinstate a post address match program status;

logic that branches the executing of the program to the third address; and

logic that executes the program having instructions stored in the memory beginning at the third address.

27. The micro-controller of claim 17, wherein the logic that invokes the address match routine is responsive to an address match interrupt service routine

having a corresponding address match interrupt entry in a vector table stored in the memory.

28. The micro-controller of claim 17, further comprising:

logic that retrieves a second address from the correction code identifying a starting address within the at least a portion of correction code;

logic that branches the executing of the program to the second address; and

logic that executes the at least a portion of the correction code beginning at the second address.

29. The micro-controller of claim 17, wherein instructions for continuing executing the program after the at least a portion of the correction code executes are included in the correction code.

30. The micro-controller of claim 17, wherein the logic that stores correction code in at least one of a plurality of correction blocks comprises:

logic that selects at least one of the plurality of correction blocks for storing the correction code;

logic that erases the selected at least one memory correction block; and

logic that transfers the correction code from an external source to the selected at least one of the plurality of correction blocks.

31. The micro-controller of claim 30, wherein the correction code is transferred from the external source by at least one of a wired and a wireless connection.

32. The micro-controller of claim 17, wherein the logic that stores correction code in at least one of a plurality of correction blocks is responsive to at least one of a detection that an external source is coupled to the memory and an invocation of a periodically scheduled maintenance routine.

33. The micro-controller of claim 17, wherein the electrically erasable programmable memory is included on at least one of a same electronic chip as the remaining logic comprising the micro-controller and a different electronic chip as the remaining logic comprising the micro-controller.

34. An electrically erasable programmable memory based memory map structure supporting an address match interrupt scheme, the memory comprising:

a plurality of correction blocks for storing correction code;

a random access memory (RAM) area including a program stack for temporarily storing program information;

a main program area for storing at least one executable program;

an initialization area for storing code to enable an address match interrupt for at least one of the correction blocks;

a special function register (SFR) area including a plurality of address interrupt registers;

a vector table for triggering the address match interrupt when a program counter associated with the at least one executable program matches a register value stored in one of the plurality of address interrupt registers; and

an interrupt service routine (ISR) area including an address match ISR;

wherein the address match ISR identifies which one of the plurality of correction blocks corresponds to the triggering of the address match interrupt to execute at least a portion of the correction code in place of at least one instruction of the at least one executable program during program execution.

35. The structure of claim 34, wherein each of the correction blocks comprise:

a first data value for determining whether correction code stored in the correction block is to be executed during program execution;

a second data value for identifying which one of the plurality of correction blocks corresponds to the triggering of the address match interrupt;

a first address identifying a location of the portion of the executable program replaced by the at least a portion of the correction code;

a second address identifying a starting address of the at least a portion of the correction code to be executed in place of the least a portion of the executable program; and

a third address identifying a return address within the executable program where program execution is continued after execution of the at a portion of the correction is completed.